

周报(2014.10.6-2014.10.12)

1. 向量场粒子动画可视化改进

1) 问题

原先使用的方法在投影和插值上有较大误差，导致结果不理想

2) 尝试解决办法

将向量场进行颜色编码，然后作为纹理覆盖在地球表面并绘制，结果保存在纹理(FBO)中。

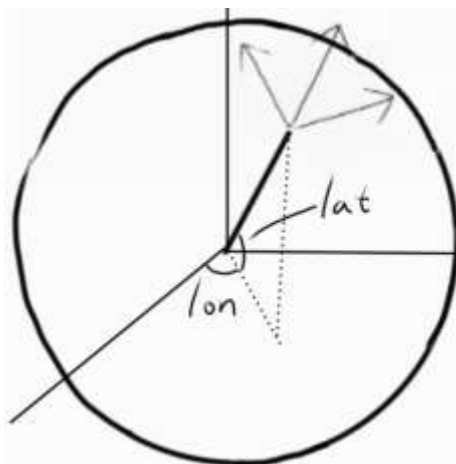
这样就完成了三维向二维的投影。但同时向量(即编码的颜色值)本身也需要做坐标系的变换。

3) 流程

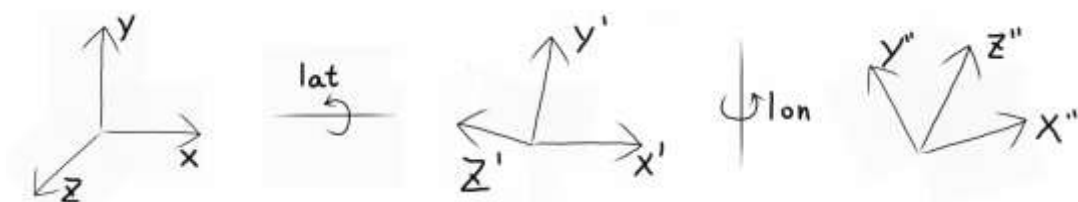
a) $(u, v) \rightarrow (x, y, z)$

风场原始数据分为 u 和 v 两个分量，分别代表在经度(东为正)和纬度(北为正)两个方向中的速度分量。需要先将其转化至绘制空间((x, y, z) 空间)中。

考虑经度为 lon ，纬度为 lat 的点，其位置和局部坐标系如下图：



考虑将坐标系从原始位置需要经过的旋转变换如下图



此外还需要一次距离为地球半径的平移，在开始前沿 z 方向或结束后沿 z'' 方向均可。前者更易以矩阵表达。通过四元组的旋转矩阵（附录.1）可以得到两次旋转的矩阵，在与平移矩阵按顺序相乘即可得到两个坐标系间的过渡矩阵。

按照公式（附录.2），求出过渡矩阵的逆即可用于进行向量在坐标系间的转化。在求逆过程中可以利用正交矩阵的逆等于其转置这一特性简化计算。

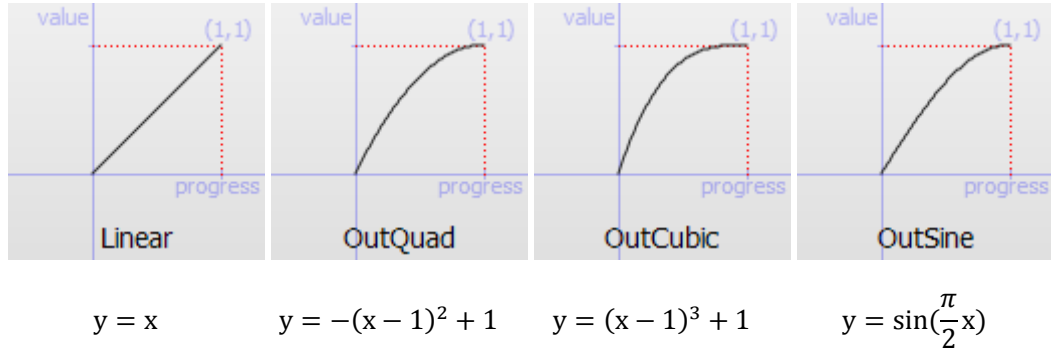
b) $(x, y, z) \rightarrow$ 屏幕空间 (x, y)

这一步可以简单的在 Shader 中完成，即将编码后的数值从纹理中取出，然后使用类似一般顶点变换的做法乘上 mvp 矩阵，最后除以其 w 分量即得到屏幕坐标，其分量范围为 $[-1, 1]$ 。需要注意的是由于输出纹理会被按照 $[0, 1]$ 范围截断，所以需要进行重参数化以将值域从 $[-1, 1]$ 变为 $[0, 1]$ 。

c) 取 FBO 中的结果纹理用于粒子动画

其他条目

- 镜头操作（拖动、缩放等）的动画直接沿时间线性分布效果不好，原因在于结束时速度不为 0，使得动画显得突兀。可以使用其他函数，如二次、三次幂函数或者三角函数等进行插值，使得终止时速度为 0（下图）。根据实际使用体验，使用 OutSine 函数进行插值时效果最好。



附录

1) 四元组旋转

向量 \mathbf{p} 沿 \mathbf{u} 为轴旋转 θ 度，其结果为

$$\mathbf{p}' = \mathbf{q}\mathbf{p}\mathbf{q}^{-1}$$

其中

$$\mathbf{q} = e^{\frac{\theta}{2}(u_x\mathbf{i}+u_y\mathbf{j}+u_z\mathbf{k})} = \cos\frac{\theta}{2}(u_x\mathbf{i} + u_y\mathbf{j} + u_z\mathbf{k})\sin\frac{\theta}{2}$$

最终可以归结为如下形式

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \mathbf{M}(\mathbf{q}) \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

其中

$$\mathbf{M}(\mathbf{q}) = \begin{bmatrix} c + u_x^2(1-c) & u_xu_y(1-c) - u_zs & u_xu_z(1-c) + u_ys \\ u_yu_x(1-c) + u_zs & c + u_y^2(1-c) & u_yu_z(1-c) - u_xs \\ u_zu_x(1-c) - u_ys & u_zu_y(1-c) + u_xs & c + u_z^2(1-c) \end{bmatrix}$$

其中

$$s = \sin\theta, c = \cos\theta$$

而且有 $\mathbf{M}(\mathbf{q})$ 是正交矩阵。

2) 坐标系变换

对于空间中两组坐标系的基有如下关系

$$\begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{pmatrix} = \mathbf{P} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{pmatrix}$$

则称 \mathbf{P} 为过渡矩阵。

对于在 α 坐标系下表达为 $(x_1, x_2, \dots, x_n)^T$ 的向量，其在 β 坐标系下表达 $(x'_1, x'_2, \dots, x'_n)^T$ 有

$$\begin{pmatrix} x'_1 \\ x'_2 \\ \vdots \\ x'_n \end{pmatrix} = \mathbf{P}^{-1} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$